Presentation of Project II (KuTanYa)

Purpose

The aim of this project is to detect spesific holes on objects in some factories with the external online camera using OpenCV library in Python programming language. It's a Project built on demand by a worker of theese kind of factory.

Why I used OpenCV? What are the adventages of it?

Generally OpenCV preffered for projects that works on live camera, because:

→ Python has several popular libraries for computer vision, including OpenCV, scikit-image , and TensorFlow. Each of these libraries has its strengths and weaknesses, and the choice of which library to use depends on the specific application and requirements.

→ OpenCV[1] is a widely-used computer vision library that provides a comprehensive set of tools for image and video processing, including feature detection, object tracking, and deep learning. It is written in C++ and has a Python interface, which makes it fast and efficient for real-time applications. OpenCV has a large community of developers, and its documentation is extensive and well-maintained.

→ scikit-image[2] is another Python library for image processing that provides a higher-level interface than OpenCV, making it easier to use for certain tasks such as image segmentation and filtering. It is built on top of NumPy and SciPy, which makes it easy to integrate with other scientific Python libraries. scikit-image is open-source and has a friendly community of developers.

→ TensorFlow[3] is a popular deep learning framework that includes a module for computer vision called TensorFlow-2.x. It provides tools for building and training neural networks for tasks such as object detection and image classification. TensorFlow has become a standard in the machine learning community, and its extensive documentation and tutorials make it easy to learn and use.

→ In summary, the choice of which library to use for computer vision depends on the specific application and requirements. OpenCV is a comprehensive library for image and video processing, scikit-image provides a higher-level interface for certain tasks, and TensorFlow is a deep learning framework that includes tools for computer vision. All of these libraries are open-source and have active communities of developers, making them excellent choices for computer vision applications in Python.

---

[1] https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html , https://docs.opencv.org/4.x/d5/d54/group__objdetect.html

[2] https://scikit-image.org

[3] https://www.tensorflow.org/?hl=tr

Why not used the face recognition library?

We can not use this Python library because of work field mean this library for only face detection and some operations about faces (like digital make-up) and that's why we can't use this library. It work as detecting par of faces – like eyes, mouth, etc.- as you can see and try from the link: https://facerecognition.readthedocs.io/en/latest/readme.html .

What is Hough Circle Transform?

The aims of it is:

→ To use Hough Transform to find circles in an image.

Theory

A circle is represented mathematically:

as $(x-xcenter)^2+(y-ycenter)^2=r^2$ where (xcenter,ycenter) is the center of the circle, and r is the radius of the circle. From equation, we can see we have 3 parameters, so we need a 3D accumulator for hough transform, which would be highly ineffective. So OpenCV uses more trickier method, **Hough Gradient Method** which uses the gradient information of edges.

We use the function: **cv.HoughCircles (image, circles, method, dp, minDist, param1 = 100, param2 = 100, minRadius = 0, maxRadius = 0)**

Parameters

image	8-bit, single-channel, grayscale input image.
circles	output vector of found circles(cv.CV_32FC3 type). Each vector is encoded as a 3-element floating-point vector (x,y,radius) .
method	detection method(see cv.HoughModes). Currently, the only implemented method is HOUGH_GRADIENT
dp	inverse ratio of the accumulator resolution to the image resolution. For example, if dp = 1 , the accumulator has the same resolution as the input image. If dp = 2 , the accumulator has half as big width and height.

minDist minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed.

param1 first method-specific parameter. In case of HOUGH_GRADIENT , it is the higher threshold of the two passed to the Canny edge detector (the lower one is twice smaller).

param2 second method-specific parameter. In case of HOUGH_GRADIENT , it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first.

minRadius minimum circle radius.

maxRadius maximum circle radius.